IN THE SPECIFICATION

In the specification, please replace the paragraphs numbered 14, 23, 24, 27, 29, 31, 32, 34, 36, 37, 38, 39, 40, 42, 43, and 44 in the original application with the following paragraphs bearing these same numbers. A marked up version of the amended portions of the specification is attached hereto as Appendix A.

[0014]   According to an embodiment of the invention, once a netlist has been generated in a standard fashion from a logic design, the implementation of that netlist onto an FPGA in an optimized fashion is automated to arrive at a set of scripts, setup files, etc. The embodiment may proceed through several iterations during the process. For example, in one embodiment of the invention, logic groups associated with the design are first initially placed on an FPGA and sorted to find general placement guidelines that meet the design constraints. Then, once the general placement for the logic groups has been formed, a resource usage estimate is obtained for the logic groups to determine how much of the target device will be used. Once the resource usage and placement have been established, the process will determine whether the proposed placement is likely to pass timing requirements associated with the design. Finally, once the design constraints, timing requirements, and resource usage targets have been satisfied, placed and routed design is tested against a test suite. If everything passes, the design is considered final and the process terminates. This process will be described in greater detail below.

[0023]   According to an embodiment of the invention, once a netlist has been generated in a standard fashion from a logic design, the implementation of that netlist onto an FPGA in an optimized fashion is automated to arrive at a set of scripts, setup files, etc. The embodiment may proceed through several iterations during the process. For example, in one embodiment of the

invention, logic groups associated with the design are first initially placed on an FPGA and sorted to find general placement guidelines that meet the design constraints. Then, once the general placement for the logic groups has been formed, a resource usage is obtained for the logic groups to determine how much of the target device will be used. Once the resource usage and placement have been established, the process will determine whether the proposed placement is likely to pass timing requirements associated with the design. Finally, once the design constraints, timing requirements, and resource usage targets have been satisfied, placed and routed design is tested against a test suite. If everything passes, the design is considered final and the process terminates. This process will be described in greater detail below.

[0024] Figs. 3A and 3B illustrate an embodiment of the invention in which source files are merged into a design for placement on a FPGA. The source files, in this embodiment, encompass the actual design such as the modes of the device, the pins, and the logic, that will be implemented in the FPGA once the design layout is completed. In Figs. 3A and 3B, Fig. 3A illustrates an example of one way of obtaining a set of design files, such as a filled netlist, a hollowed netlist, design constraints, and data-path constraints. The netlists and constraint sets are defined by the physical structure of the FPGA and the logic to be implemented in it. The invention, one example of which is illustrated in Fig. 3B, takes this information and transforms it into a program that may be actually used and downloaded to an FPGA.

[0027] By synthesizing 111, 112 the original and hollowed sources 100, 103, two netlists are created: a hollowed netlist 115 and a filled netlist 114. The filled netlist 114 contains the full implementation of the design as described in the original source files 100, but the hollowed netlist 115 only contains the boundaries of the defined logical groups with no internal control

logic. In a similar way, two sets of constraints are created: data-path constraints 120 and design

constraints 118. The design constraints 118 represent the design specifications 104 as originally

intended, where the data-path constraints 120 only describe specifications for signals between

logical groups.

[0029]   Although Fig. 3A has been described in some detail to explain one possible method

of obtaining a set of hollowed netlists 102 (extracted from the original source files), filled netlists

114 (created by the RTL synthesis), constraints (such as the data-path constraints 118 and design

constraints 120), and test environment 128, the invention is not limited to this described

environment as numerous other environments may be used to create these files as well. Thus,

the invention is not limited to this embodiment as the invention may be used in many different

forms to merge the information obtained during this or a similar process to completion of an

FPGA design.

[0031]   As shown in Fig. 3B, the software includes several different stages, which are

configured to perform different functions in the automated FPGA design process.   The first

stage, referred to herein as "initial placement 140" is configured to perform initial placement of

the logic groups on the FPGA. This enables the logic groups to be placed on the FPGA without

worrying about their resource usage, and without consideration as to whether this placement will

satisfy the timing requirements. Having a preliminary placement is advantageous in that the

logic groups may be sorted out and arranged to be close to the pins that they will control, and to

be close to other logic groups with which they will frequently interact.

[0032]   In one embodiment, the initial placement (also referred to herein as architectural

analysis) is performed in the software by using the hollowed netlists and the design constraints to

see if the initial iteration can be executed without any errors, and to see if the design constraints syntax is correct. If the FPGA does not pass using the initial placement, the input files are altered and re-tested until an initial placement on the FPGA is obtained. During this process, timing constraints are ignored and logic groups are mapped. By using a hollowed netlist instead of a filled netlist, the initial placement may be performed rapidly since the files being manipulated are relatively small.

[0034]   As shown in Fig. 3B, one way of performing the initial placement process is to combine the hollowed netlists and design constraints 142. This combination is then checked to see if it passes 144, i.e., if the constraints specified in the constraints files can be read and there are no errors in the hollowed netlist. If not, the source files are modified and the process iterated until no errors are found.

[0036]   Once there is an initial placement, the next step is to evaluate the resource usage of the logic groups and what specific device should be used to implement the FPGA design. Determining the specific device to be used may involve selecting one of several FPGAs in a family or may involve selecting a device from between different FPGA families. FPGA families are generally produced by a manufacturer with the same basic architecture but with different numbers of basic FPGA elements. Often it is desirable to be able to select the FPGA with the smallest number of basic elements for a given design since that is likely to be the least expensive from a manufacturing standpoint. Determining the resource usage of each logic group in the design, enables an FPGA of an appropriate size to be selected.

[0037]   Fig. 4B illustrates one example of how sizing the logic groups can affect the design of the FPGA. As shown in Fig. 4B, during the sizing process several of the blocks may increase or

decrease in size. Additionally, as shown in Fig. 4C, an FPGA from the FPGA family was selected on which the design may be implemented. This FPGA had slightly less resources than the FPGA used initially in Figs. 4A-4B. The relative placement of the logic groups may change in subsequent parts of the automated FPGA design process; however, the initial placement should not be dramatically affected by the resizing of the logic groups within a same FPGA family. Optionally, the process may iterate through the initial placement 140 and resource usage estimate 160 procedures until a design meeting these two criteria is obtained (as shown in Fig. 4D).

[0038]   According to one embodiment of the invention, as shown in Fig. 3B, the logic groups are sized by running the filled netlists with the data-path constraints 162 and the result tested 164 to see if the initial logic group sizes were adequate and/or optimal given the filled netlists and data-path constraints. For example, one of the initial logic group size may be too big and another too small. The sizes and shapes may be adjusted until the appropriate sized logic groups are determined.

[0039]   Once appropriately sized logic groups are obtained, it may be desirable to include a margin of error at this point such that the logic groups should not use more than 70% of the available basic elements on the FPGA. If the number of basic elements used by the logic groups is much higher than 70% of the available basic elements, it may be desirable to use a larger FPGA in the FPGA family. If the number of basic elements is much smaller than 70%, it may be desirable to use a smaller FPGA in the FPGA family. The invention is not limited to a 70% margin as other margin numbers may be used without departing from the invention. For example, one way of selecting the proper FPGA is to iterate the design through parts in the same

FPGA family 166 and check to see if the logic groups in the design occupy a target percentage (such as 70%) of the available basic elements for that FPGA family member (168); where the usage value is too far off of the target percentage, another family member may be selected. Ultimately, the specific FPGA device and resource usage estimates will be returned (170) to be used by other parts of the automated FPGA design process.

[0040]   Once the specific FPGA is selected and the sizes of the logic groups are known, the software performs a timing analysis 180 to ensure the design will meet the timing requirements, it may be desirable to include a margin of error (also referred to herein as slack) at this point such that timing requirements are met with a specified slack. In this process, the hollowed netlist is run with the logic groups properly sized and placed on the FPGA, and timing delays are optimized in view of the data-path timing constraints. The hollowed netlist is used at this point, so the reported slack is merely an estimate. However, performing a timing slack analysis may enable the placement, shape, and/or resource usage of the logic groups to be adjusted to enable timing issues to be alleviated. Additionally, by using hollowed netlists at this point, the timing estimate may be performed very rapidly. Optionally, where the slack requirements significantly alter the FPGA design, the first three processes (initial placement, resource usage estimation, and timing estimation) may be iterated until an acceptable design is found that satisfies all three processes. Additionally, the shape of the logic groups and the value of timing constraints may be adjusted at this stage to enable the logic groups to better utilize the available basic elements on the FPGA.

[0042] According to one embodiment, timing estimate may be performed by combining the hollowed netlists, resource usage estimates and data-path constraints and testing 184 to see if the

area groups (which may have been rearranged during the resource usage estimation process) still satisfy the data-path timing requirements. If not, the process iterates until it does. Subsequently, the process checks the timing characteristics 186 of the design and iterates until the predetermined slack is achieved 188. If timing specifications are not being met or if the desirable slack is not achieved, it may be desirable to use an FPGA with a faster speed grade in the FPGA family. If timing specifications are being met with relatively large slack, it may be desirable to use an FPGA with a slower speed grade in the FPGA family. Any timing margin may be selected depending on the implementation and the function to which the FPGA will be used. In the example, a 10% timing margin is suggested, although the invention is not limited to this embodiment. The invention is not limited to use of a slack figure of about a 10%, as other margin numbers may be used without departing from the invention.

[0043]   Finally, once the timing analysis 180 has been completed, final placement 200 is performed and the improved design is tested to see if it performs as expected. Specifically, at this stage, the logic groups have been placed, sized, and adjusted into area groups that meet all constraints. They are now filled and the logic inserted into the area groups, and the final design is tested to see if it operates in accordance with expectations and within margins defined in the test suite (e.g. trivial test bench 128). One example of how the process of filling the area groups may be implemented is illustrated in Fig. 4F.

[0044]   According to one embodiment of the invention, for example as shown in the embodiment illustrated in Fig. 3B, the filled netlists are merged with the area groups 202 and the design constraints to then be analyzed to see if it passes 204. If not, the process iterates by altering the placement, size, or other constraints to obtain a completed design. Once the process

has completed, the filled design is run with the design constraints 206 and a design environment

is created 208. This generates the scripts, setup files, tool lineup files, etc., 210 that will

ultimately be used to program the FPGA.